10/23/00

A

PTO/SB/05 (08-00)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this box → ⊞

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.	CR9-97-092-US2
First Inventor	Kasichainula
Title	Method & Apparatus for Remotely...
Express Mail Label No.	EK873466515US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. [X] Fee Transmittal Form (e.g., PTO/SB/17) *(Submit an original and a duplicate for fee processing)*
2. [ ] Applicant claims small entity status. See 37 CFR 1.27.
3. [X] Specification [*Total Pages* 42] *(preferred arrangement set forth below)*
   - Descriptive title of the invention
   - Cross Reference to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to sequence listing, a table, or a computer program listing appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings *(if filed)*
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
4. [X] Drawing(s) *(35 U.S.C. 113)* [*Total Sheets* 7]
5. Oath or Declaration [*Total Pages* 2]
   a. [ ] Newly executed (original or copy)
   b [X] Copy from a prior application (37 CFR 1.63 (d)) *(for continuation/divisional with Box 17 completed)*
      i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b)
6. [ ] Application Data Sheet. See 37 CFR 1.76
7. [ ] CD-ROM or CD-R in duplicate, large table or Computer Program *(Appendix)*
8. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. [ ] Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. [ ] CD-ROM or CD-R (2 copies); or
      ii. [ ] paper
   c. [ ] Statements verifying identity of above copies

ADDRESS TO:

Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

ACCOMPANYING APPLICATION PARTS

9. [ ] Assignment Papers (cover sheet & document(s))
10. [ ] 37 CFR 3.73(b) Statement *(when there is an assignee)* [ ] Power of Attorney
11. [ ] English Translation Document *(if applicable)*
12. [X] Information Disclosure Statement (IDS)/PTO-1449 [ ] Copies of IDS Citations
13. [X] Preliminary Amendment
14. [X] Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
15. [ ] Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
16. [ ] Other: .....

17. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:*

[ ] Continuation [X] Divisional [ ] Continuation-in-part (CIP) of prior application No: 08/931,979

Prior application information Examiner S. Lao Group / Art Unit: 2755

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS

[ ] *Customer Number or Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or [X] *Correspondence address below*

Name	Jeanine S. Ray-Yarletts				
Address	IBM Corp., Dept. T81/Bldg. 503-3				
	P.O. Box 12195				
City	Research Triangle Park	*State*	NC	*Zip Code*	27709
Country	U.S.A.	*Telephone*	919-543-2541	*Fax*	919-254-4330

Name (Print/Type)	Jeanine S. Ray-Yarletts	*Registration No. (Attorney/Agent)*	39,808
Signature	Jeanine S. Ray-Yarletts	*Date*	Oct. 20, 2000

Burden Hour Statement This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231 DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

Docket CR9-97-092-US2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Kasichainula et al

Serial No.:

Filed: Herewith

For: Method and Apparatus for Remotely Running Objects Using Data Streams and/or Complex Parameters

Assistant Commissioner of Patents
Washington, D.C. 20231

EXPRESS MAIL CERTIFICATE

WE REQUEST THE DATE OF DEPOSIT AS THE DATE FILED.

I hereby certify that the following **attached** correspondence comprising:

1. Utility Patent Application Transmittal for Divisional Application, in duplicate
2. Application, 42 pages
3. Drawings, 7 sheets
4. Oath and Declaration, copy as filed in parent application
5. Information Disclosure Statement, USPTO Form 1449 and one Reference
6. Preliminary Amendment
7. Business Reply Post Card

is being deposited with the United States Postal Service Express Mail Post Office to Addressee service under CFR 1.10 on the date indicated above and is addressed to:

Box Patent Application
Assistant Commissioner of Patents
Washington, D.C. 20231

Dianne Lane
(Name of person mailing paper or fee)

Dianne Lane
Signature of person mailing paper or fee)

IN THE UNITED STATES PATENT & TRADEMARK OFFICE

In re application of	:	October 20, 2000
M.V. Kasichainula et al	:	IBM Corporation
Ser. No.	:	Dept.T81/Bldg. 503-3
Filed Herewith	:	P.O. Box 12195
For: Method & Apparatus for Remotely	:	Res. Tri. Park, NC 27709
Running Objects Using Data Streams and/or	:	Art Unit: 2755
Complex Parameters	:	Examiner: S. Lao

Preliminary Amendment

Assistant Commissioner for Patents
Washington, DC 20231

Sir:

This preliminary amendment is filed with divisional application of parent application serial number 08/931,979 as a response to election/restriction requirement. Please enter the following amendments before examination of the present application.

In the Specification:

After the title, insert:

– Cross Reference to Related Application

This application is a divisional of U.S. Application No. 08/931,979 filed September 17, 1997. –

In the Claims:

Cancel Claims 1-15 without prejudice.

Remarks

Applicant has filed herewith a divisional application to parent application serial number 08/931,979. Claims 1-27 remained in that Application. Applicant has hereby canceled Claims 1-15. Claims 16-27 remain in the divisional application filed herewith.

Respectfully submitted,

Jeanine S. Ray-Yarletts
Attorney for Applicants
Reg. No. 39,808

JSR:jdl

Phone: 919-543-2541
Fax: 919-254-4330

EXPRESS MAIL LABEL NO.: _____ DATE OF DEPOSIT: _____

I hereby certify that this paper and fee are being deposited with the United States Postal Service Express Mail Post Office to Addressee service under 37 CFR §1.10 on the date indicated above and is addressed to the Assistant Commissioner of Patents, Washington, D.C. 20231.

NAME OF PERSON MAILING PAPER AND FEE _____ SIGNATURE OF PERSON MAILING PAPER AND FEE _____

INVENTORS: Manoj V. S. Kasichainula, Zhiyong Li

Method and Apparatus for Remotely Running Objects using Data Streams and/or Complex Parameters

Field of the Invention

The present invention relates generally to computer programming methods and systems, and, in particular, to object oriented programming and to methods and systems of running object oriented programs on multiple computers connected by a network.

Background of the Invention

The distribution of a single program, so that a portion of the program executes on more than one computer, has become more pervasive as "desktop"

computers have become more powerful. Figure 1 demonstrates an example of a computer network containing several distributed computers upon which an application can execute. While most computer networks are many orders of magnitude larger, this small network is used as an example. Presently, many computer systems allow objects to communicate over a network. One method of distributing object oriented programs across computer networks is Automatic Object Distribution (AOD), described in the commonly assigned, copending patent application having serial number 08/852,263, entitled "A Process for Running Objects Remotely," filed on May 6, 1997.

When a distributed object-oriented program is built with AOD, it is written as if the entire system is to reside on a single machine and compiled into unlinked executable code, known as "byte code." The distribution of the objects is determined, and the AOD process is used to effect the distribution by analyzing the aforementioned byte code, determining which method calls will be made across the computer network, and generating proxy objects to represent remote objects and their method calls. Two proxies are created for each object-to-object call that will occur over the network: one that resides on the machine containing the object making the call, and one that resides on the machine containing the object to which the call is made. These two proxies cooperate to hide the fact that the objects actually reside on different machines from the programmer, thereby sparing the programmer any need to be aware of the distributed nature of the system when writing his code.

When calls are made across a computer network in this fashion, the parameters are objects. When these objects represent finite data, such as arrays, data structures, integers, etc. they are passed using well-known techniques such as Object Serialization. In Object Serialization, all the pieces of the object are combined and converted to a byte stream, which is sent across the network and then reassembled into a new data object on the other side of the network connection. This method is well-known in the art, and is used by the AOD system to send parameters for method calls which are split across the network.

However, some types of data objects do not lend themselves to Object Serialization. In particular, objects which represent data streams and complex objects (which may contain extensive programmed methods and/or references to other objects, have system locking requirements, or do not implement a serialization protocol) do not lend themselves easily to object serialization.

Objects which represent data streams are not a finite size, as they often represent data to be read from or written to some data source, such as a diskette drive. Additionally, since they may represent data being input to or output from a system or peripheral device, the data they contain may be in a system-specific format. When a program is distributed with AOD, different parts of it may be distributed to different types of computers, which may have incompatible data formats.

Additionally, under the AOD proxy system, if one data stream object is split and proxied, all data streams must be so split and proxied, because the proxy remaining on the first machine would contain method names and semantics that are identical to those in a real data stream, causing name collisions. These name collisions would eliminate the possibility of any kind of data input or output operations on a machine on which a data stream proxy resides, a result which is not acceptable in today's environment. If the name collision problem were to be overcome, the performance of the proxied data stream would not be acceptable, as each read or write on a data stream so proxied would require two method calls and a remote method call.

When complex objects are passed as parameters on remote method calls, it is undesirable to pass them as simpler objects would be passed, using serialization, for three reasons:

1. A complex object will likely contain extensive code and/or references to other objects. To serialize the object, all of the data it contains and the objects it references must be serialized and passed over the network as a byte stream. This may be inefficient, especially if the object is being passed to allow access to only a small percentage of its data.

2. When a complex object is passed over the network to be used by another object, it must be locked on its "home" machine. This is to prevent any other objects from invoking it, which may result in changes to the data contained therein, while it is being operated on in a remote machine. Without such

locking, the object may become corrupted. For example, if an object contains a counter, and the object is invoked on a remote machine to increment the counter and also invoked on the local machine to increment the same counter, then when the remote copy is returned and recopied over the original copy, the counter's value will be one less than required. Locking imposes a performance penalty on all objects that need to access the object being serialized, as they are all required to queue up and wait for the remote operation on the object to complete and for the results to be returned and recopied.

3. A complex object that is to be serialized must contain special code to enable such serialization. For example, in the Java™ (Java is a trademark of Sun Microsystems) environment, the object must implement the serializable interface. This may require the programmer to be aware of and code for the distributed nature of the program.

Object of the Invention

It is an object of the present invention to provide a method for creating distributed object programs, which include data streams and complex objects as parameters, that allows the programmer to write a program as if it were running on a single, local machine.

It is a further object of the present invention to automatically create the distribution of data streams across multiple machines even though the format of stream data may be machine-specific.

It is yet a further object of the present invention to allow data streams that are local to a machine and those which have been distributed to other machines to co-exist without programmer input.

Summary of the Invention

The present invention provides a system, method and program product for executing or running objects remotely, some of which objects may pass data streams between themselves. This is herein referred to as Proxy Datastream Handling (PDH). A data stream is an object which represents a source of data, such as a diskette drive, which is accessed to obtain data from the source. Additionally, the present invention provides a system, method and program product for executing or running objects remotely, some of which objects may pass complex objects between themselves. This is herein referred to as Complex Object Parameter Handling (COPH). A complex object is one which contains programmed member functions and/or references to other objects, or which has locking requirements to the extent that it is not desirable to pass the object itself over a network.

As shown in Figure 2, an object oriented program may be written with many independent objects that call each other to perform a unified function. Figure

2 depicts a computer memory 101 in which objects 102 V, W, X, Y & Z all reside. One or more of these objects may pass or receive a data stream or a complex object as a parameter. The present invention allows a user to move some of the objects from a first computer to a second computer to be executed. This is shown in Figure 3. In Figure 3 objects V & Y 205 remain in a first computer memory 201 while objects W, X, & Z 207 are moved to a second computer memory 203. This can be done, for instance, to balance the workload between the first computer and the second computer. This creates additional complexities since each of the program objects are capable of calling external routines herein referred to as methods. If one or more of the methods references one or more data streams as parameters, the format of the data produced therefrom may be machine-specific. Additionally, if one or more of the methods references one or more complex objects, additional complexities are introduced, e.g., the complex object may have locking requirements. For example, a complex object may be updatable by many different objects. Sending a complex object to a remote machine to make it available for an updating method requires all of the other objects that may need to update it to be locked out for the entire duration of the remote method call, thereby affecting system performance. Also, a complex object may contain extensive data fields and referenced objects which make it undesirable to pass the entire complex object over the network simply to allow a possibly minor aspect of the object to be used by the remote object.

The separation of the program objects into multiple computers for execution requires that all of the methods or objects which, due to the separation of the

objects to different computer memories now access methods and data streams which are no longer coresident with the calling method on the same system, know that some of the objects and data streams are located elsewhere on the network and understand how to access those remote objects.

The Proxy Datastream Handling (PDH) and the Complex Object Parameter Handling (COPH) of the present invention eliminate the requirement that programmers write their programs with the knowledge that the objects which reference data streams or pass complex objects as parameters will be distributed over a network. It also eliminates the need for programmers to write programs in a special language, known as an Interface Definition Language (IDL), that supports distributed objects, or to write using any particular tool when it is desired to distribute the work across the network. Instead, using the present invention, programmers write programs in a common object-oriented language, such as the Java™ language, and compile the code into unlinked executable code. (In the computing literature, unlinked executable code is often called "object code". When describing object-oriented systems, the use of the term "object code" creates confusion, therefore the term "bytecodes" has been adopted for purposes of the present invention.) The programmer writes the code exactly as if writing a program that will execute on a single machine; he does not decorate the program with any additional information as he would with IDL systems. The programmer then decides where the distribution should occur, then PDH and COPH, together with Automatic Object Distribution (AOD), generate the distribution code.

The AOD performs the distribution by generating two proxies. The proxies allow method calls written for local invocation to be invoked over a network. If a split is indicated between some class Y to be executed locally, and some class X to be executed remotely, the AOD process generates the proxies to overcome the intervening network. Together, these proxies intercept the calls from Y, pass them to X, and return the result to Y. If a data stream is passed as a parameter to these calls, the present invention enhances the proxies to do the PDH necessary to obtain data from the original data stream object, send the data that the data stream object contains over the network, and reassemble it in the destination machine into a proxy data stream object that can be accessed as if it were local. This is shown in more detail in Figures 4A and 4B. If a complex object is passed as a parameter to these calls, the present invention enhances the proxies to do the COPH necessary to build proxies for the complex object and reference the complex object's proxies on the actual call. This is shown in more detail in Figures 5A and 5B.

Data Streams:

Figure 4A represents the computer system in which a program is running in a single memory 401. Both object Y 404 and object X 403 reside on the same memory and a method call 407 is used to invoke object Y 404 from object X 403. The result of this invocation is that a reference to a data stream object is returned 417 to object X 403, and object X 403 may use the reference to access this data stream object 422 via a method call 423 to obtain the data 424 that the data stream object 422 represents.

As shown in Figure 4B, the AOD process uses information in the bytecode file for a class, say Y 404, to generate the two proxy files (Y' 405 and Y"406) that contain classes serving as proxies for Y 404. Y' 405 contains a class named Y that has all of the public methods in Y 404. (Note that since the original class Y 404 file will be located on the remote machine, no name conflict exists.) Y" 406 contains a class with a unique name that contains the method that makes calls to the public methods in class Y. Y ' 405 resides on the same machine as X 403; Y" 406 resides on the same machine as Y 404. Since Y' 405 contains a class called Y and that class contains all of the public methods in Y 404, when X 403 makes a call against a method in the original class Y 404, that call is actually made against a method in the new class Y residing locally and contained in file Y' 405. Y' 405 passes the call information to Y" 406 which then makes a local call to Y 404. The results of the local call are propagated back to X 403 by way of Y" 406 and Y' 405. When the call returns a data stream, Y" 406 and Y' 405 cooperate to communicate the data that the data stream represents across the network and present it to the caller as if it were local data. The result is that the application, written to execute locally, can be distributed across the network without requiring that the programmer write or change any of the program code.

Complex Objects:

Figure 5A represents a computer system in which a program is running in a single memory 501. Object Y 503, object X 502, and object Z 504 all reside on the same memory and a method call 505 is used to invoke object Y 503

from object X 502. Object Z 504 is a parameter of the method call, and as part of the method call object Y 503 performs a method call to invoke an object which is passed to object Y 503 as a parameter (in this case, object Z 504). Object Y 503 updates object Z 504 because object Z is provided to object Y 503 by object X 502 as a parameter. Object X 502 could provide itself, or any other object that contains the method that object Y 503 will use in its call. The case in which object X provides itself as a parameter, indicating that it is to be invoked by object Y, is referred to hereafter as a callback. This callback logic may be used in an object oriented system when the invocation of object Y 503 from object X 502 may cause system states which are stored in object X to be altered, or if invoking methods in the calling object is the way that the called object returns its results. The non-callback logic, in which object X 502 provides object Y 503 with a third object to be updated, such as object Z 504, may be used when object Z is a common system object for storing state information or other data that may be updated by object Y.

Figure 5B represents the computer system of figure 5A after it has been distributed using the AOD process described above. Objects X 502 and Z 504 remain in computer 501, but object Y 503 has been moved to another computer 509. The AOD process knows from examining the objects' byte codes that object X 502 will be calling object Y 503, so it creates two proxies for object Y, one on computer 501 referred to as Y' 510 and one on computer 509 referred to as Y" 511. Because the AOD process also knows that object Z 504 will be passed as a parameter on the call from X 502 to Y 503, the proxies for object Y are enhanced by the present invention to create the

appropriate proxies for object Z 504 so that object Z may be passed as a parameter on the remote call to object Y without need to lock and serialize object Z. Additionally, since the logic is added by the AOD process to the proxies created for object Y, no programmer intervention is necessary to accomplish this efficient remote call using object Z, allowing the entire program to be written as if all its objects are to reside on one computer.

Brief Description of the Drawings

The preferred embodiment of the present invention will be herein described in more detail with reference to the drawings, in which:

Figure 1 depicts a sample network upon which the present invention may appear.

Figure 2 depicts a computer memory containing five objects.

Figure 3 depicts a distributed system upon which two objects execute in one computer memory and three objects execute in a different computer memory.

Figure 4A depicts a method where two objects execute in the same memory, and a data stream is a returned result from a method call

Figure 4B depicts the objects of Figure 4A after being distributed using AOD and the present invention.

Figure 5A depicts a method where two objects execute in the same memory, and a third object is passed as a parameter to a method call

Figure 5B depicts the objects of Figure 5A after being distributed using AOD and the present invention.

Detailed Description of the Preferred Embodiment

The present invention is applicable to any automated distributed programming utilizing proxies and data streams and using complex objects as parameters on remote calls. Proxy Datastream Handling (PDH) and complex object handling of the preferred embodiment uses the Java™ programming environment although the present invention is not limited to Java™ environments and the application of this invention to other object-oriented environments would be straightforward to one skilled in the art. The preferred embodiment of the present invention assumes that the programmer has written a program in the Java™ language as if it were to run locally (e.g., it does not include any Remote Method Invocation code), and used the standard Java™ compiler to compile the source files into corresponding bytecode files. The Automatic Object Distribution (AOD) process is used to distribute the objects and their proxies into the network, said proxies having been enhanced by the present invention to handle data streams and/or complex objects that may be parameters of one or more of the method calls to be made across the network.

Data Streams:

Figure 4B depicts the system of figure 4A after being distributed by AOD between two computers, the client 401 and the server 402 systems. Object X 403 is a client object and object Y 404 is a server object. To allow X and Y to be automatically distributed, proxy objects are created for Y in both computers. In the client computer 401, a proxy Y' 405 is created which has the same name and method names as Y 404, so that X 403 may make calls as if Y 404 were local. Y' 405 then translates the calls to semantics of Y" 406 and calls Y" in the server computer 402 .

In the server computer, a proxy Y" 406 is created which receives calls from Y' 405, translates them to the semantics that Y 404 requires and invokes the desired methods in Y 404.

When the client object X 403 is calling the server object Y 404 to obtain stream data, the proxies implement the process of the present invention.

In figure 4B, X 403 is shown calling a method of Y 404 to read a stream of data from an input source such as diskette. (The input source may be any source of data that may be represented by a data stream object.) The call 407 is actually made to proxy object Y' 405, which passes the call via Remote Method Invocation (RMI) or some other standard remote calling method 408 to Y" 406. Y" then translates the call into the semantics that the actual object, Y 404, requires and then passes the call to it. Y returns the result 410 in the form of

a reference to a data stream object 418, which is accessed to obtain the data in the steam.

Upon receipt of the data stream object reference, Y" creates 411 a thread 412 which creates a network connection 414 with Y' 405 and then continuously reads data 419 and 420 using the data stream object and sends that data 413 out on the connection 414 to Y' as raw data (not as a data stream object). Upon creating this thread and passing the data stream to it, the method in Y" has completed its work and returns. Although in this preferred embodiment the network connection is created by the thread which Y" creates, the connection could also be created by Y'.

Upon creation 421 of the network connection 414, proxy object Y' 405 understands that the connection is for data that is to be returned via a data stream object, and builds the data stream object such that when it is accessed to obtain data 423, the data 424 is read 415 from the socket connection. (In the preferred Java™ embodiment, a system call 416 may be used to build the data stream object, however other methods may be used in the preferred or other embodiments.) Y' then returns 417 this locally-created data stream object 422 to X 403. Thus X receives the requested data stream object and accesses it as if its source were local to machine 401, even though the data is actually local to machine 402 and being provided via a network connection 414.

This preferred embodiment describes the process taken when the data source resides on the server and is returned to the client. However, a similar process

would be used if the data source resides on the client 401 and is being provided as a parameter to the server 402. For example, the data source could represent a keyboard on the client computer on which a user inputs requests or other data. In that case,Y' 405 would create the thread to retrieve stream data and place it on the network connection, and Y" 406 would create the data stream object to read the data from the network connection.

There are other configurations that could implement the present invention. For example, there may be more than one data stream to be passed on a call, in which case the process described above would be used for each data stream, either by creating one thread and network connection per data stream to be passed, or by passing multiple data stream objects over fewer threads and/or connections, a technique known as multiplexing. Methods for performing this multiplexing are well-known in the art, and information can be found at http://java.sun.com/products/jdk/1.1/docs/guide/rmi/spec/rmi-protocol.doc.html#3477, for example. Additionally, there may be data streams flowing in both directions on a call (i.e., both returned on the call and passed as parameters), in which case each proxy object would implement the invention in both directions, both building one or more threads to read local stream data onto one or more network connections and building one or more local data stream objects to receive data being sent over one or more network connections from the other side.

Complex Objects

Figure 5B depicts the system of figure 5A after being distributed by AOD between two computers, the client 501 and the server 509 systems. Object X 502 is a client object and object Y 503 is a server object. To allow X and Y to be automatically distributed, proxy objects are created for Y in both computers. In the client computer 501, a proxy Y' 510 is created which has the same name and method names as Y 503, so that X 502 may make calls as if Y 503 were local. Y' 510 then translates the calls to semantics of Y" 511 and calls Y" in the server computer 509 .

In the server computer, a proxy Y" 511 is created which receives calls from Y' 510, translates them to the semantics that Y 503 requires and invokes the desired methods in Y.

When a complex object, such as object Z 504, is used as a parameter on the remote method call from the client object X 502 to the server object Y 503, the proxies implement the process of the present invention.

In Figure 5B, Object X 502 is shown making a remote method call to object Y 503, and object Z 504 is one of the parameters. The call 505 is actually made to proxy object Y' 510. Before passing the call via Remote Method Invocation (RMI) or some other standard remote calling method to Y" 511, object Y' examines the call and realizes that complex object Z 504 is one of the parameters. Without the process of the present invention, object Y' 510

would simply serialize object Z and lock it, and send it to object Y" 511. Serialization would require that all of object Z 504, including all its data and objects referenced by it, be sent over the network to object Y" 511. Serialization may be inefficient, as the amount of data to be sent over the network may be very large, particularly if object Z is a large and/or very complex object. Locking would be necessary to prevent other objects on machine 501 from making changes to object Z 504 while it is being operated on in machine 509. Without locking, object Z 504 could become corrupted. For example object Y 503 may update a counter in its copy of object Z. Before the copy of object Z is returned to machine 501, another object in machine 501 may update the same counter in the actual object Z residing on machine 501. When object Y on machine 509 completes its operation on its copy of object Z, the object must be recopied to machine 501, and after the recopy object Z's counter would be one less than the correct value. The required locking may severely degrade performance on machine 501, as objects that need to update object Z 504 would all be required to queue up, waiting for the remote call to complete before they can perform their required updates.

Using the process of the present invention, object Y' 510 creates a proxy object Z" 513 for object Z 504 before passing the call to Y" 511. This proxy contains the code necessary to allow a reference to itself to be passed over the network (for example, in the Java™ environment it implements the serializable interface). Object Y' 510 also sets up a reference table entry 524 in which the key Z" returns an indication of object Z 504. A reference table is defined as any data structure or representation into which data can be placed and

retrieved using keys, for example a hash table. Then, object Y' passes the call to object Y" via Remote Method Invocation or some other standard remote calling method 514, and a reference to Z" is provided in place of Z as the parameter in the call.

Upon receiving the remote call from Y' 510, Y" 511 creates a proxy Z' 512 for object Z 504 in machine 509, and creates 515 a reference table entry in which the key Z' returns a remote call reference to the proxy Z" 513 which was created by Y' 510. Then when object Y" 511 translates the call into the semantics of object Y 503 and invokes object Y, a reference to proxy object Z' 512 is passed as the parameter. Thus object Y will invoke object Z' 512 when object Y's invoked method invokes the object passed in as a parameter.

When object Z' 512 is invoked 506 by object Y 503, object Z' uses the reference table entry which was created earlier 515 by object Y" 511 to determine where the call is to be directed. By looking itself up in the table 517, object Z' receives a reference to object Z" 513 on machine 501. Using the received reference, object Z' translates the call into the semantics of object Z" and invokes object Z" using Remote Method Invocation or some other standard remote calling method 518.

When object Z" 518 is invoked, it uses the reference table entry which was created earlier 524 by object Y' to determine where the call is to be directed. By looking itself up in the table 519, object Z" receives a reference to object Z 504. Using the received reference, object Z" translates the received call into

the semantics of object Z and invokes object Z. Thus, object Y 503 has successfully invoked object Z in the course of its invocation from object X 502, even though object Y resides on a different computer than objects X and Z.

Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns said result to object Y 503. Then when object Y finishes the method which was invoked from object X 502, it returns the result 523, if any, of said invocation to object Y" 511, which returns said result to object Y' 510, which returns said result to object X 502, this completing the object X's method invocation to object Y 503, which also updated object Z 504. The proxies do all the work so that the original objects X, Y, and Z may be programmed as if they all reside on one computer.

This preferred embodiment describes the case in which the complex object is passed as a parameter on a remote call. A similar process would be employed if the complex object is returned by the remote callee to the caller. For example, the call may return a reference to a system-wide status object which actually resides on the server computer.

What is claimed is:

1. A computer implemented method for distributing objects of a program, each object containing one or more programmed member functions, said member functions passing and/or returning one or more data stream objects as parameters between at least two physical devices, said method comprising the computer executable steps of:

identifying all of the objects in said program;

determining which of said objects are to reside on a first computer and which of said objects are to reside on a second computer such that the resulting distributed system comprises at least a first object on a first computer and a second object and said one or more data stream objects on a second computer;

identifying all programmed member functions that may be accessed from a remote computer;

generating a first proxy and a second proxy for each said object that may be accessed from a remote computer, said first proxy residing on said first computer and said second proxy residing on said second computer, said first proxy containing network linkage and indication to access programmed member functions on said second proxy residing on said second computer

including logic to transfer said one or more data stream objects and said second proxy containing linkage and indication to access said programmed methods on said second object including logic to transfer said one or more data stream objects; and,

accessing said remote programmed member functions through said first and second proxies.

2. A computer implemented method as claimed in Claim 1 wherein:

-- said logic in said second proxy object in said second computer to transfer one or more data stream objects to said first proxy object in said first computer comprises the computer executable steps of:

creating or accepting one or more network connections with said first proxy object in said first computer;

accessing each said data stream object to obtain the data each said data stream object represents; and,

sending, via said one or more network connections, said data to said first proxy residing on said first computer.

-- said logic in said first proxy object in said first computer to transfer one or more data stream objects from said second proxy in said second computer comprises the computer executable steps of:

accepting or creating said one or more network connections with said second proxy;

creating a quantity of data stream objects equal to the quantity of data stream objects to be transferred, each said created data stream object to represent data received on said one or more network connections; and,

providing access reference to each of said created data stream objects to said first object on said first computer.

3. A method as claimed in Claim 2, wherein data stream objects exist on both said first computer and said second computer, and said method is performed bidirectionally.

4. A method as claimed in Claim 2 or Claim 3, wherein one network connection is created for each data stream object to be transferred.

5. A method as claimed in Claim 2 or Claim 3, wherein fewer network connections than data stream objects are created, and said data is distributed to said more than one data stream objects created on said first computer via multiplexing.

6. A computer program product for distributing objects of a program, each object containing one or more programmed member functions, said member functions passing and/or returning one or more data stream objects as parameters, across at least two physical devices, said computer program product comprising:

a computer-readable storage medium having computer-readable program code means embodied in said medium, said computer-readable program code means comprising:

computer-readable program code means for identifying all of the objects in said program;

computer-readable program code means for determining which of said objects are to reside on a first computer and which of said objects are to reside on a second computer such that the resulting distributed system comprises at least a first object on a first computer and a second object and said one or more data stream objects on a second computer;

computer-readable program code means for identifying all programmed member functions that may be accessed from a remote computer;

computer-readable program code means for generating a first proxy and a second proxy for each said object that may be accessed from a remote computer, said first proxy residing on said first computer and said second proxy residing on said second computer, said first proxy containing network linkage and indication to access programmed member functions on said second proxy residing on said second computer including logic to transfer said one or more data stream objects and said second proxy containing linkage and indication to access said programmed methods on said second object including logic to transfer said one or more data stream objects; and,

computer-readable program code means for accessing said remote programmed member functions through said first and second proxies.

7. A computer program product according to Claim 6 wherein:

-- said logic in said second proxy object in said second computer to transfer one or more data stream objects to said first proxy object in said first computer comprises the computer executable steps of:

creating or accepting one or more network connections with said first proxy object in said first computer;

accessing each said data stream object to obtain the data each said data stream object represents; and,

sending, via said one or more network connections, said data to said first proxy residing on said first computer.

-- said logic in said first proxy object in said first computer to transfer one or more data stream objects from said second proxy in said second computer comprises the computer executable steps of:

accepting or creating said one or more network connections with said second proxy;

creating a quantity of data stream objects equal to the quantity of data stream objects to be transferred, each said created data stream object to represent data received on said one or more network connections; and,

providing access reference to each of said created data stream objects to said first object on said first computer.

8. A computer program product as claimed in Claim 7, wherein data stream objects exist on both said first computer and said second computer, and said method is performed bidirectionally.

9. A computer program product as claimed in Claim 6 or Claim 7, wherein one network connection is created for each data stream object to be transferred.

10. A computer program product as claimed in Claim 6 or Claim 7, wherein fewer network connections than data stream objects are created, and said data is distributed to said one or more data stream objects created on said first computer via multiplexing.

11. A computer system for distributing objects of a program, each object containing one or more programmed member functions, said member functions passing one or more data stream objects as parameters, across more than one physical device, said system comprising:

means for identifying all of the objects in the program;

means for determining which of the objects are to reside on a first computer and which of the objects are to reside on a second computer such that the resulting distributed system comprises at least a first object on a first computer and a second object and said one or more data stream objects on a second computer;

means for identifying all programmed member functions that may be accessed from a remote computer;

means for generating a first proxy and a second proxy for each object that may be accessed from a remote computer, said first proxy residing on said first computer and said second proxy residing on said second computer, said first proxy containing network linkage and indication to access programmed member functions on said second proxy residing on said second computer including logic to transfer said one or more data stream objects and said second proxy containing linkage and indication to access said programmed

methods on said second object including logic to transfer said one or more data stream objects; and,

means for accessing said remote programmed member functions through said first and second proxies.

12. A system as claimed in Claim 11 wherein:

-- said logic in said second proxy object in said second computer to transfer one or more data stream objects to said first proxy object in said first computer comprises the computer executable steps of:

creating or accepting one or more network connections with said first proxy object in said first computer;

accessing each said data stream object to obtain the data each said data stream object represents; and,

sending, via said one or more network connections, said data to said first proxy residing on said first computer.

-- said logic in said first proxy object in said first computer to transfer one or more data stream objects from said second proxy in said second computer comprises the computer executable steps of

accepting or creating said one or more network connections with said second proxy;

creating a quantity of data stream objects equal to the quantity of data stream objects to be transferred, each said created data stream object to represent data received on said one or more network connections; and,

providing access reference to each of said created data stream objects to said first object on said first computer.

13. A system as claimed in Claim 12 wherein data stream objects exist on both said first computer and said second computer, and said method is performed bidirectionally.

14. A system as claimed in Claim 11 or Claim 12, wherein one network connection is created for each data stream object to be transferred.

15. A system as claimed in Claim 11 or Claim 12, wherein fewer network connections than data stream objects are created, and said data is distributed to said one or more data stream objects created on said first computer via multiplexing.

16. A computer implemented method for distributing one or more objects of a program across more than one physical device, each object containing one or more programmed member functions, said member functions having

complex objects, said complex objects including one or more programmed member functions, as parameters, said method comprising the computer executable steps of:

identifying all of the objects in the program;

determining which of the objects are to reside on a first computer and which of the objects are to reside on a second computer such that the distributed system will consist of at least a first object on a first computer and a second object on a second computer;

identifying all programmed methods contained in each object that may be accessed from a remote computer;

generating a first proxy and a second proxy for each object that may be accessed from a remote computer, said first proxy residing on said first computer and said second proxy residing on said second computer, said first proxy containing network linkage and indication to access programmed member functions on said second proxy on said second computer including logic to transfer and translate complex objects which reside on said first computer used as member function parameters and said second proxy containing linkage and indication to access said programmed member functions on said second object including logic to transfer and translate complex objects, said complex objects containing one or more programmed member functions and reside on said first computer, used as member function parameters; and,

accessing said remote programmed methods through said proxies.

17. A method as claimed in Claim 16, wherein:

said logic in said first proxy on said first computer to transfer and translate complex data objects comprising the steps of:

creating a third proxy, for said complex object, which is to reside on said first computer with said complex object, said third proxy containing linkage and indication to access programmed member functions on said complex object;

creating a reference table entry which correlates said third proxy object to said complex object, which may be accessed by said third proxy object to access said complex object; and,

passing as a member function parameter to said second proxy on said second machine a reference to said third proxy, in place of said complex object when said complex object is to be a parameter in a member function call to said second object on said second machine.

said logic in said second proxy on said second computer to transfer and translate complex data objects comprising the steps of:

creating a fourth proxy for said complex object on said first computer which is to reside on said second computer, said fourth proxy

containing network linkage and indication necessary to access programmed member functions on said third proxy on said first machine;

creating a reference table entry which correlates said fourth proxy to a reference to said third proxy on said third computer, which may be accessed by said fourth proxy to access said third proxy;

passing as a member function parameter to said second object from said second proxy on said second computer an indication of said fourth proxy, in place of said reference to said third proxy on said first computer, which represents said complex object on said first computer.

said network linkage and indication in said fourth proxy necessary to access programmed member functions on said third proxy on said first computer comprising the steps of:

looking up said fourth proxy in said reference table on said second computer to determine which object on said first machine said fourth object is a proxy for, said lookup returning a reference to said third proxy on said first computer;

calling the appropriate programmed member functions in said third proxy on said first computer.

said linkage and indication in said third proxy necessary to access programmed methods on said complex object comprising the steps of:

looking up said third proxy in said reference table on said first computer to determine which object on said first machine said third object is a proxy for, said lookup returning a reference to said complex object on said first computer;

calling the appropriate programmed member functions in said complex object.

18. A method as claimed in Claim 17 wherein one of said complex objects is said first object on said first computer.

19. A method as claimed in Claim 17 wherein said reference table is a database.

20. A computer program product for distributing one or more objects of a program across more than one physical device, each object containing one or more programmed member functions, said member functions having complex objects, said complex objects including one or more programmed member functions, as parameters, said computer program product comprising:

a computer-readable storage medium have computer-readable program code means embodied in said medium, said computer-readable program code means comprising:

computer-readable program code means for identifying all of the objects in the program;

computer-readable program code means for determining which of the objects are to reside on a first computer and which of the objects are to reside on a second computer such that the distributed system will consist of at least a first object on a first computer and a second object on a second computer;

computer-readable program code means for identifying all programmed methods contained in each object that may be accessed from a remote computer;

computer-readable program code means for generating a first proxy and a second proxy for each object that may be accessed from a remote computer, said first proxy residing on said first computer and said second proxy residing on said second computer, said first proxy containing network linkage and indication to access programmed member functions on said second proxy on said second computer including logic to transfer and translate complex objects which reside on said first computer used as member function parameters and said second proxy containing linkage and indication to access

said programmed member functions on said second object including logic to transfer and translate complex objects, said complex objects containing one or more programmed member functions and reside on said first computer, used as member function parameters; and,

computer-readable program code means for accessing said remote programmed methods through said proxies.

21. A computer program product as claimed in Claim 20, wherein:

said logic in said first proxy on said first computer to transfer and translate complex data objects comprising the steps of:

creating a third proxy, for said complex object, which is to reside on said first computer with said complex object, said third proxy containing linkage and indication to access programmed member functions on said complex object;

creating a reference table entry which correlates said third proxy object to said complex object, which may be accessed by said third proxy object to access said complex object; and,

passing as a member function parameter to said second proxy on said second machine a reference to said third proxy, in place of said complex

object when said complex object is to be a parameter in a member function call to said second object on said second machine.

said logic in said second proxy on said second computer to transfer and translate complex data objects comprising the steps of:

creating a fourth proxy for said complex object on said first computer which is to reside on said second computer, said fourth proxy containing network linkage and indication necessary to access programmed member functions on said third proxy on said first machine;

creating a reference table entry which correlates said fourth proxy to a reference to said third proxy on said third computer, which may be accessed by said fourth proxy to access said third proxy;

passing as a member function parameter to said second object from said second proxy on said second computer an indication of said fourth proxy, in place of said reference to said third proxy on said first computer, which represents said complex object on said first computer.

said network linkage and indication in said fourth proxy necessary to access programmed member functions on said third proxy on said first computer comprising the steps of:

looking up said fourth proxy in said reference table on said second computer to determine which object on said first machine said fourth object is

a proxy for, said lookup returning a reference to said third proxy on said first computer;

calling the appropriate programmed member functions in said third proxy on said first computer.

said linkage and indication in said third proxy necessary to access programmed methods on said complex object comprising the steps of:

looking up said third proxy in said reference table on said first computer to determine which object on said first machine said third object is a proxy for, said lookup returning a reference to said complex object on said first computer;

calling the appropriate programmed member functions in said complex object.

22. A computer program product as claimed in Claim 21 wherein one of said complex objects is said first object on said first computer.

23. A computer program product as claimed in Claim 21 wherein said reference table is a database.

24. A computer system for distributing one or more objects of a program across more than one physical device, each object containing one or more

programmed member functions, said member functions having complex objects, said complex objects including one or more programmed member functions, as parameters, said system comprising:

means for identifying all of the objects in the program;

means for determining which of the objects are to reside on a first computer and which of the objects are to reside on a second computer such that the distributed system will consist of at least a first object on a first computer and a second object on a second computer;

means for identifying all programmed methods contained in each object that may be accessed from a remote computer;

means for generating a first proxy and a second proxy for each object that may be accessed from a remote computer, said first proxy residing on said first computer and said second proxy residing on said second computer, said first proxy containing network linkage and indication to access programmed member functions on said second proxy on said second computer including logic to transfer and translate complex objects which reside on said first computer used as member function parameters and said second proxy containing linkage and indication to access said programmed member functions on said second object including logic to transfer and translate complex objects, said complex objects containing one or more programmed

member functions and reside on said first computer, used as member function parameters; and,

means for accessing said remote programmed methods through said proxies.

25. A system claimed in Claim 24, wherein:

said logic in said first proxy on said first computer to transfer and translate complex data objects comprising the steps of:

creating a third proxy, for said complex object, which is to reside on said first computer with said complex object, said third proxy containing linkage and indication to access programmed member functions on said complex object;

creating a reference table entry which correlates said third proxy object to said complex object, which may be accessed by said third proxy object to access said complex object; and,

passing as a member function parameter to said second proxy on said second machine a reference to said third proxy, in place of said complex object when said complex object is to be a parameter in a member function call to said second object on said second machine.

said logic in said second proxy on said second computer to transfer and translate complex data objects comprising the steps of:

creating a fourth proxy for said complex object on said first computer which is to reside on said second computer, said fourth proxy containing network linkage and indication necessary to access programmed member functions on said third proxy on said first machine;

creating a reference table entry which correlates said fourth proxy to a reference to said third proxy on said third computer, which may be accessed by said fourth proxy to access said third proxy;

passing as a member function parameter to said second object from said second proxy on said second computer an indication of said fourth proxy, in place of said reference to said third proxy on said first computer, which represents said complex object on said first computer.

said network linkage and indication in said fourth proxy necessary to access programmed member functions on said third proxy on said first computer comprising the steps of:

looking up said fourth proxy in said reference table on said second computer to determine which object on said first machine said fourth object is a proxy for, said lookup returning a reference to said third proxy on said first computer;

calling the appropriate programmed member functions in said third proxy on said first computer.

said linkage and indication in said third proxy necessary to access programmed methods on said complex object comprising the steps of:

looking up said third proxy in said reference table on said first computer to determine which object on said first machine said third object is a proxy for, said lookup returning a reference to said complex object on said first computer;

calling the appropriate programmed member functions in said complex object.

26. A system as claimed in Claim 25 wherein one of said complex objects is said first object on said first computer.

27. A system as claimed in Claim 25 wherein said reference table is a database.

Abstract

Proxy data stream handling and complex object parameter handling allow object oriented programs to be run as distributed programs without any explicit networking code, and without using an interface definition language (IDL). Two proxies are generated dynamically that allow method calls written for local invocation to be invoked over a network. These dynamically-generated proxies allow calls to flow across a network as if they were local, and contain support for using data stream and complex objects as parameters.

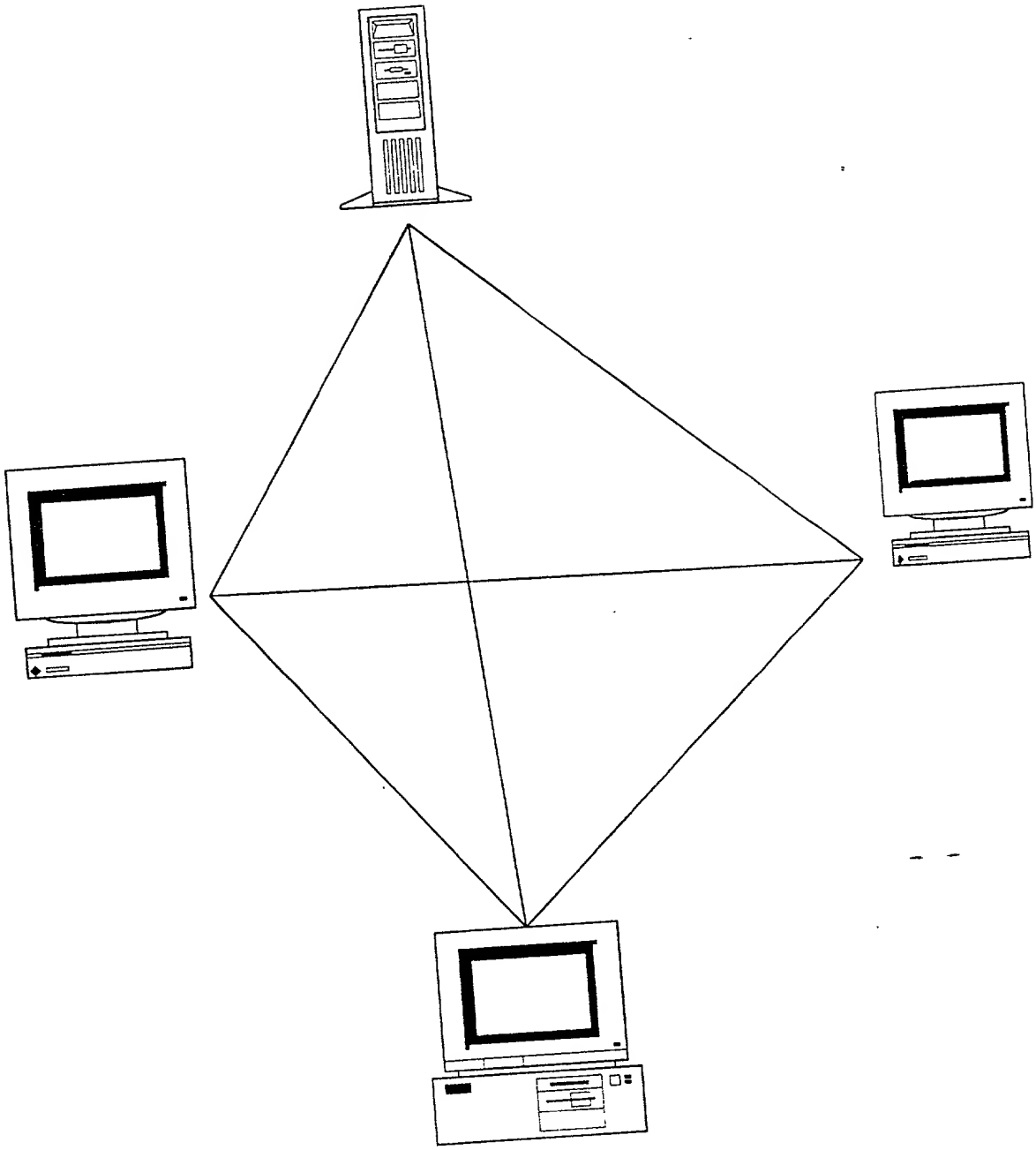

Fig. 1

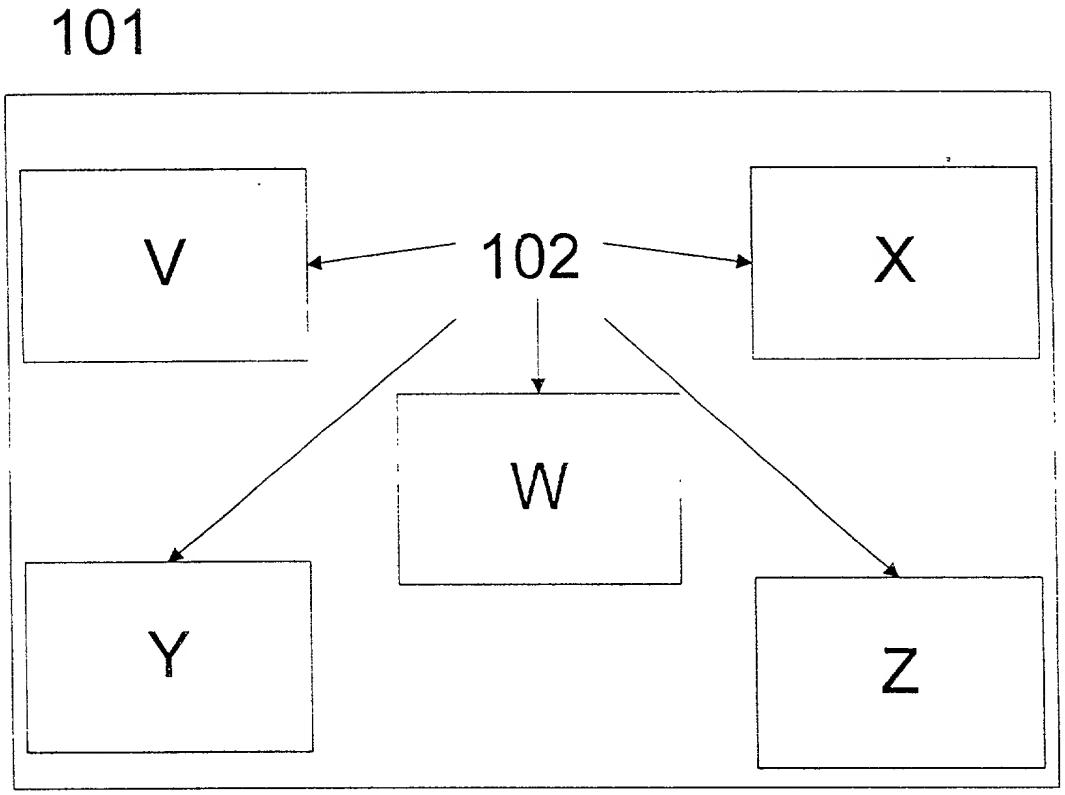

Fig. 2

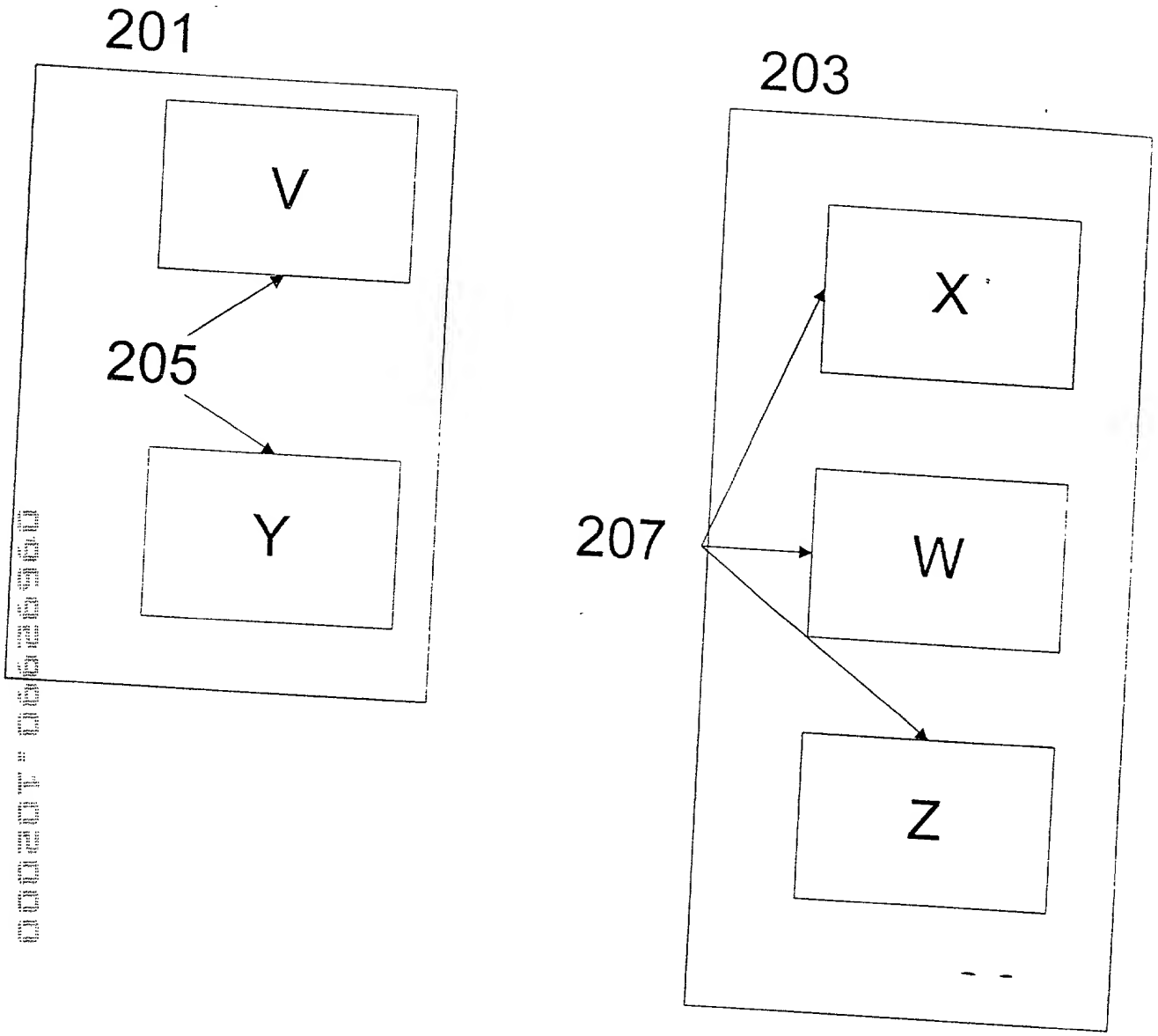

Fig. 3

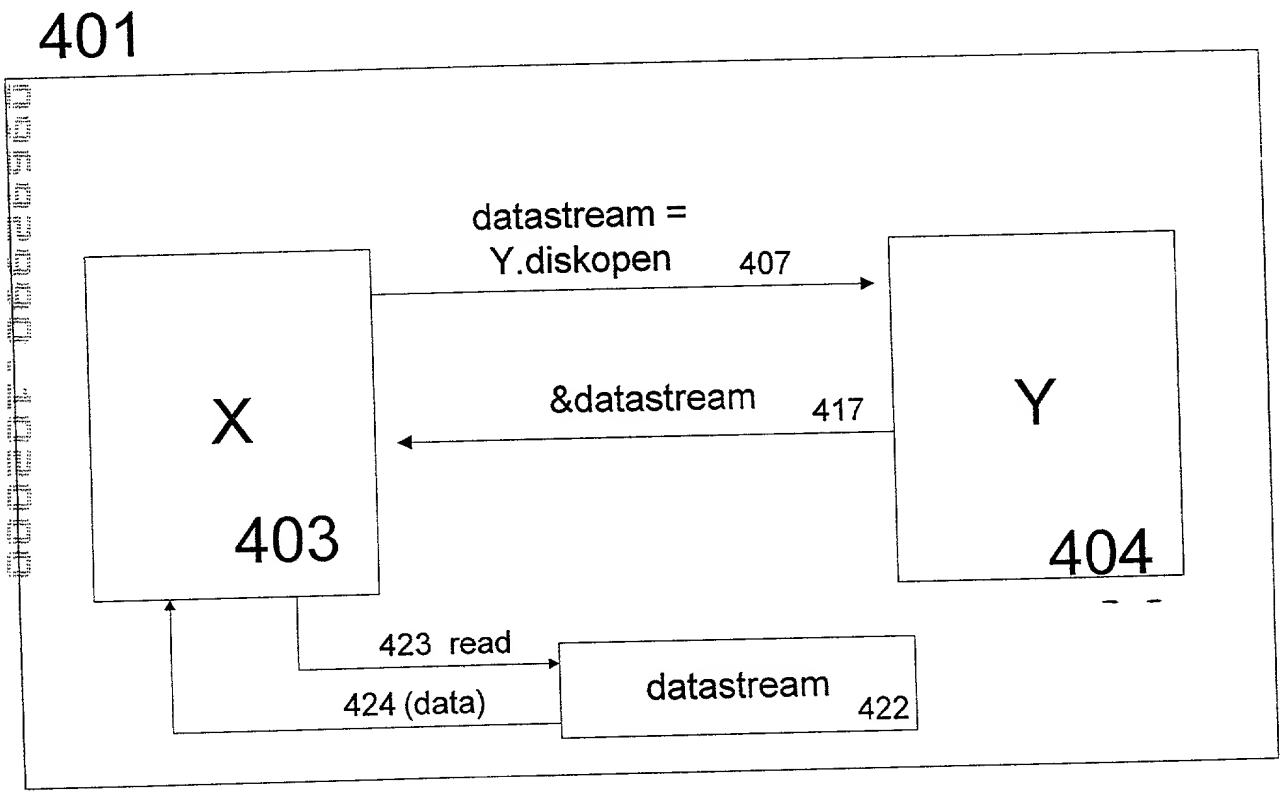
401
datastream =
Y.diskopen
407
X
403
&datastream
417
Y
404
423 read
424 (data)
datastream
422

Fig. 4A

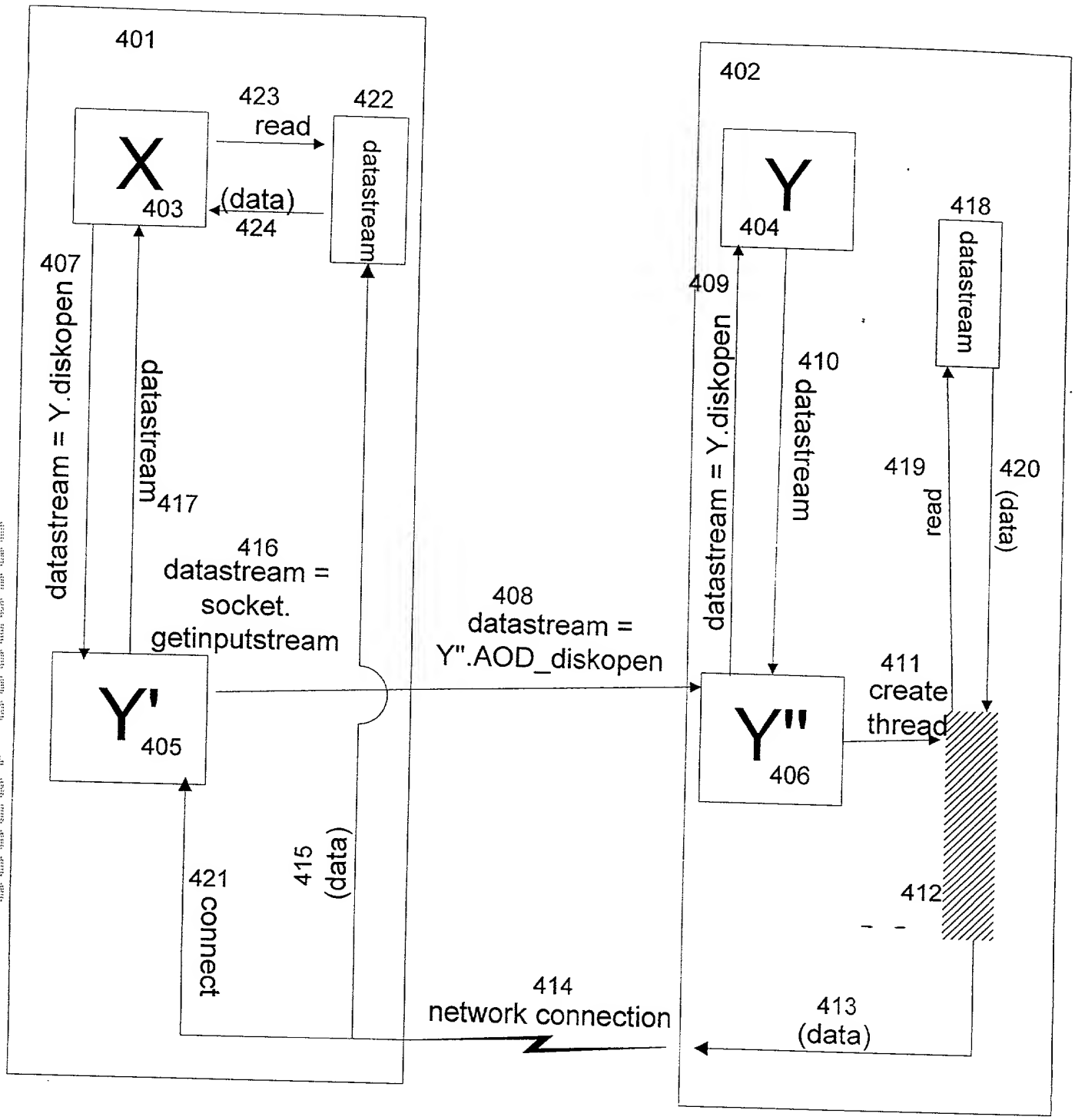
401
402
X
403
423
read
422
datastream
(data)
424
Y
404
418
datastream
407
datastream = Y.diskopen
409
datastream = Y.diskopen
410
datastream
419
read
420
(data)
datastream
417
416
datastream =
socket.
getinputstream
408
datastream =
Y".AOD_diskopen
411
create
thread
Y'
405
Y"
406
421
connect
415
(data)
412
414
network connection
413
(data)

Fig. 4B

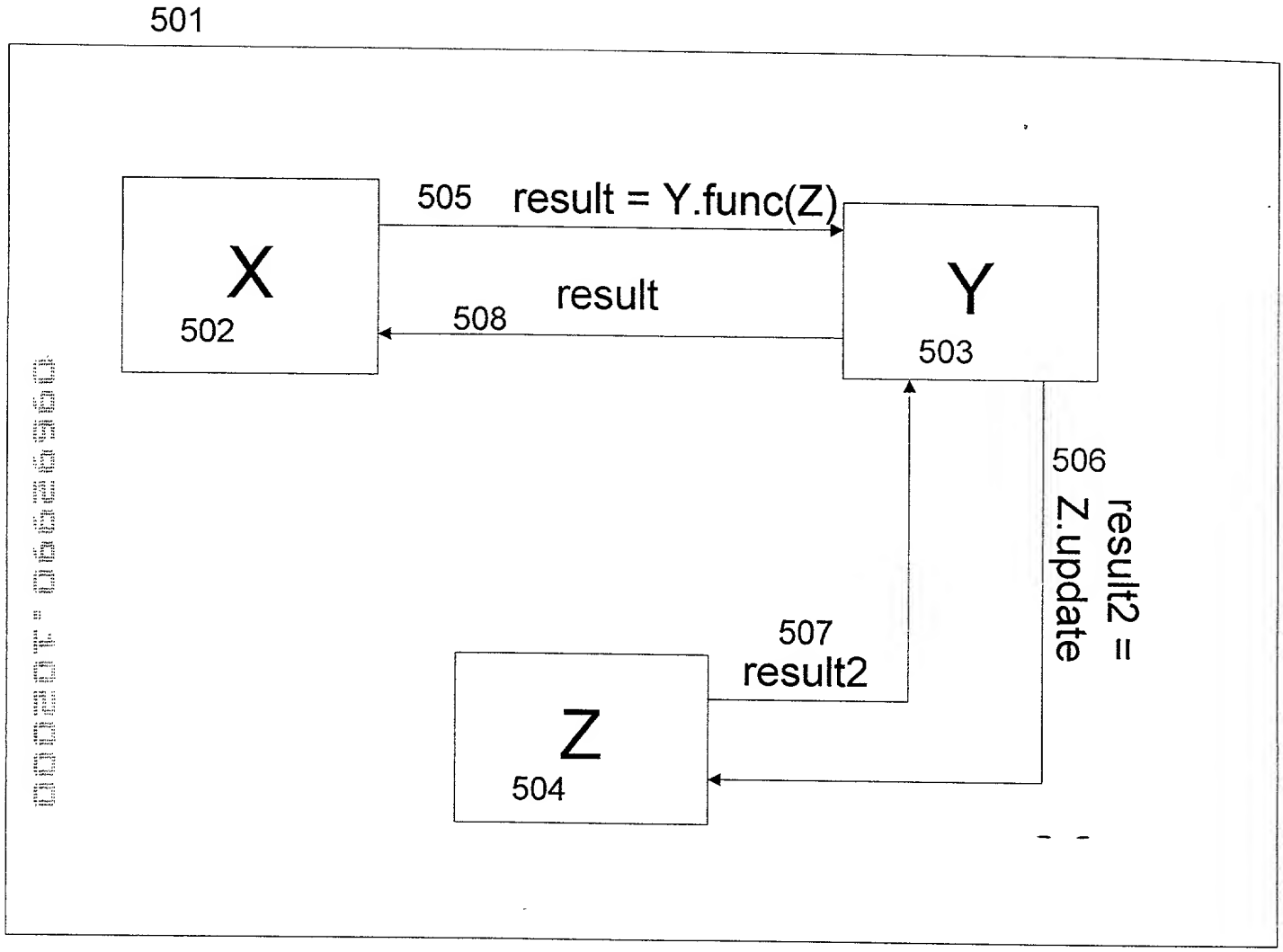
501
X
502
505 result = Y.func(Z)
result
508
Y
503
506 result2 = Z.update
507 result2
Z
504

Fig. 5A

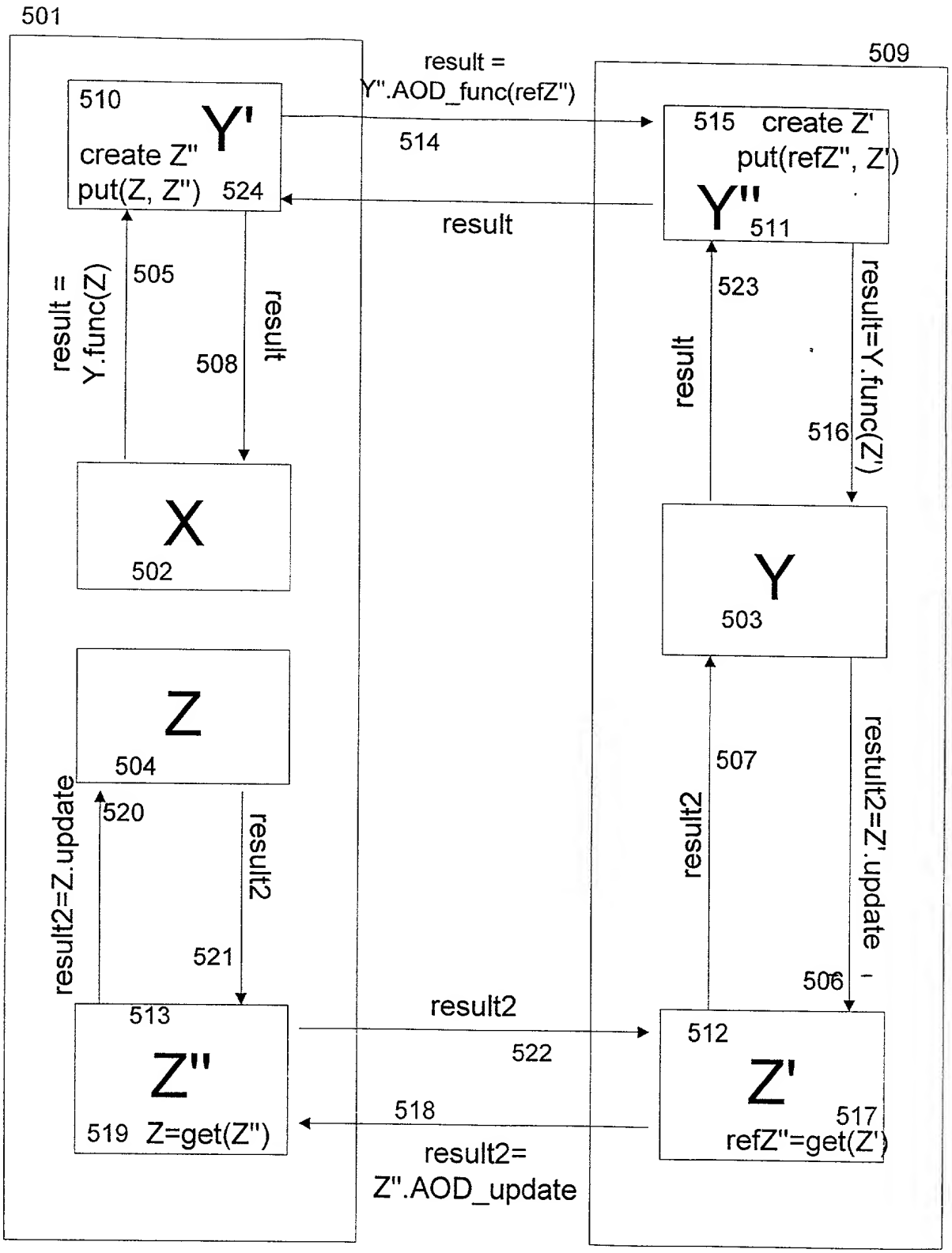
501
509
result =
Y".AOD_func(refZ")
514
510
Y'
create Z"
put(Z, Z")
524
515 create Z'
put(refZ", Z')
Y"
511
result
result =
Y.func(Z)
505
508
result
523
result
result=Y.func(Z')
516
X
502
Y
503
Z
504
520
result2=Z.update
result2
521
507
result2
result2=Z'.update
506
513
Z"
519 Z=get(Z")
result2
522
512
Z'
517
refZ"=get(Z')
518
result2=
Z".AOD_update

Fig 5B

Declaration and Power of Attorney for Patent Application

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name; I believe I am the original, first and sole inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled

Method and Apparatus for Remotely Running Objects Using Data Streams and/or Complex Parameters

the specification of which (check one)

☑ is attached hereto.

☐ was filed on _____ as Application Serial No. _____ and was amended on _____.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

Number	Country	Day/Month/Year	Priority Claimed

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Prior U.S. Applications:

Serial No.	Filing Date	Status

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both,

under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: J. S. Ray - Yarletts, Reg. No. 39.808; A. B. Clay, Reg. No. 32,121; G. M. Doudnikoff, Reg. No. 32,847; E. H. Duffield, Reg. No. 25.970; J. W. Herndon, Reg. No. 27,901; C. A. Hughes, Reg. No. 26,914; E. A. Pennington, Reg. No. 32,588; J. E. Hoel, Reg. No. 26,279; J. C. Redmond, Jr., Reg. No. 18,753.

Send all correspondence to: Jeanine S. Ray-Yarletts
IBM Corporation
Dept. T81/Bldg. 062
P.O. Box 12195
Research Triangle Park, NC 27709

---

(1) Inventor: Manoj V. S. Kasichainula

Signature: Manoj V. S. Kasichainula 9-16-97
Date

Residence: 8117 Green Lantern Street, Apt. 306
Raleigh, North Carolina 27613

Citizenship: USA

Post Office Address: Same

(2) Inventor: Zhiyong Li

Signature: Zhiyong Li 9-16-97
Date

Residence: 863 Louise Circle
Durham, North Carolina 27613

Citizenship: P. R. China

Post Office Address: Same